BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Aktenzeichen: 102 54 657.6

Anmeldetag: 22. November 2002

Anmelder/Inhaber: Philips Intellectual Property & Standards GmbH, Hamburg/DE

(vormals: Philips Corporate Intellectual Property GmbH)

Bezeichnung: Mikrocontroller und zugeordnetes Verfahren zum Abarbeiten der Programmierung des Mikrocontrollers

IPC: G 06 F 12/14

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 23. September 2003
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Ersig

A 9161
06/00
EDV-L

BESCHREIBUNG

Mikrocontroller und zugeordnetes Verfahren zum Abarbeiten der Programmierung des Mikrocontrollers

Die vorliegende Erfindung betrifft einen Mikrocontroller, dessen Programmierung in mindestens einer maschinenabhängigen Assemblersprache vorgenommen ist, bei der die Assemblerbefehle mit Ausnahme bedingter Programmsprünge bzw. konditioneller Programmverzweigungen im wesentlichen unabhängig von Daten ausführbar sind.

Die vorliegende Erfindung betrifft des weiteren ein Verfahren zum Abarbeiten der in mindestens einer maschinenabhängigen Assemblersprache vorgenommenen Programmierung eines Mikrocontrollers der vorgenannten Art.

Als Mikrocontroller werden Einchip-Mikrocomputer bezeichnet, die in der Regel zum Steuern von Geräten verwendet werden und bei denen C[entral]P[rocessing]U[nit], Speicher und Ports auf einem Chip integriert sind. Die Programmierung von Mikrocontrollern erfolgt in maschinenabhängiger Assemblersprache. Bei den bekannten Assemblersprachen werden hierbei mit Ausnahme der bedingten Programmsprünge bzw. der konditionellen Programmverzweigungen alle Assemblerbefehle unabhängig von Daten ausgeführt.

Ein bedingter Programmsprung bzw. eine konditionelle Programmverzweigung wird üblicherweise wie folgt realisiert: Die zu überprüfende Bedingung, in der Regel mindestens ein Statusflag, wird getestet. Wird hierbei festgestellt, dass ein Sprung bzw. eine Verzweigung stattfinden soll, so wird der Programmcounter mit einer neuen Programmadresse (= mit einem neuen "Wert") geladen. Wenn kein Sprung bzw. keine Verzweigung stattfinden soll, so wird die Instruktion beendet, denn der Programmcounter enthält ja schon automatisch den nächsten Wert, das heißt die nächste Adresse.

Ein derartiges Procedere bedingt, dass bei bedingten Programmsprüngen bzw. bei konditionellen Programmverzweigungen ein zeitlicher Unterschied in der Instruktionsausführung auftreten kann. Dieser zeitliche Unterschied in der Instruktionsausführung liegt darin begründet, dass im springenden bzw. verzweigenden Falle zusätzlich noch der Programmcounter auf einen neuen Wert (auf eine neue Programmadresse) gesetzt wird, während im nicht-springenden bzw. nicht-verzweigenden Falle die Instruktion nach dem Bedingungstest beendet ist.

Dies bedeutet, dass die Befehlsausführung bedingter Sprünge bzw. konditioneller Verzweigungen in Mikrocontrollerprogrammen üblicherweise unterschiedliche Ausführungszeiten und damit auch unterschiedliche, anhand dynamischer Strommessungen ermittelbare Stromwerte aufweist, und zwar in Abhängigkeit davon, ob ein bedingter Sprung bzw. eine konditionelle Verzweigung ausgeführt wird oder nicht.

Eine gängige, auch einen Missbrauch durch Angreifer ermöglichende Methode zur Softwareanalyse, zum Beispiel zum Ermitteln von kryptographischen Schlüsseln, besteht darin, mittels einer speziellen Timinganalyse bedingte Programmsprünge bzw. konditionelle Programmverzweigungen zu identifizieren und anhand des identifizierten Programmablaufs Rückschlüsse auf die verarbeiteten Daten zu gewinnen.

Damit lassen sich allein mittels des zeitlichen Ablaufs der bedingten Sprunginstruktion bzw. der konditionellen Verzweigungsinstruktion Rückschlüsse auf die in dieser Instruktion getesteten Daten ziehen, was beispielsweise bei einem unbefugten Angriff auf besonders sicherheitssensible Abschnitte eines Mikrocontrollerprogramms, wie etwa auf einen kryptographischen Schlüssel, von extremem Nachteil ist.

Bei der Implementierung von Software, die auf einem Mikrocontroller Aktionen ausführt, die für einen unbefugten Beobachter verborgen bleiben sollen, besteht also - abstrahiert formuliert - ein wesentliches Problem darin, daß der unbefugte Beobachter mittels physikalischer Messungen Informationen über den ausgeführten Code und über die hierbei benutzten Daten erhalten kann. Diese Problematik tritt insbesondere bei sicherheitsrelevanter Software auf, wie sie zum Beispiel in SmartCards verwendet wird.

Typische Versuche, Informationen über den ausgeführten Code und über die hierbei benutzten Daten zu erhalten, bestehen in Messungen an der Strom- und/oder Spannungsversorgung des Mikrocontrollers. Informationen über den internen Programmablauf können aber auch mit anderen physikalischen Meßverfahren gewonnen werden.

Vor dem Hintergrund der vorstehend geschilderten Möglichkeiten, das intern auf einem Mikrocontroller ablaufende Programm auszuspionieren, erscheint die Möglichkeit einer Verschleierung dieses Programms erstrebenswert. Konventionellerweise bekannt sind bisher allerdings lediglich zufällige Variationen hinsichtlich der einzelnen ausgeführten Instruktion, nicht jedoch hinsichtlich größerer Programmteile, so daß eine Verschleierung des intern auf dem Mikrocontroller ablaufenden Programms nur bedingt bis gar nicht möglich ist.

Ausgehend von den vorstehend dargelegten Nachteilen und Unzulänglichkeiten sowie unter Würdigung des umrissenen Standes der Technik (= vollständig reproduzierbares Abarbeiten des auf einem Mikrocontroller ablaufenden Programms in Abhängigkeit von den zu verarbeitenden Daten) liegt der vorliegenden Erfindung die Aufgabe zugrunde, einen Mikrocontroller der eingangs genannten Art sowie ein Verfahren der eingangs genannten Art so weiterzuentwickeln, dass das auf dem Mikrocontroller ablaufende Programm für einen äußeren Beobachter völlig geheim und unvorhersehbar, das heißt nicht reproduzierbar ist.

Diese Aufgabe wird durch einen Mikrocontroller mit den im Anspruch 1 angegebenen Merkmalen sowie durch ein Verfahren mit den im Anspruch 5 angegebenen Merkmalen gelöst. Vorteilhafte Ausgestaltungen und zweckmäßige Weiterbildungen der vorliegenden Erfindung sind in den jeweiligen Unteransprüchen gekennzeichnet.

Mithin ist die Lehre der vorliegenden Erfindung in einem zufallsgesteuerten Ablauf der Programmierung im Mikrocontroller zu sehen. Dies bedeutet, daß es durch geeignetes Verarbeiten von mittels mindestens eines Zufallszahlengenerators erzeugten Zufallszahlen möglich ist, ein auf dem Mikrocontroller ablaufendes Programm für einen äußeren Beobachter unvorhersehbar und nicht reproduzierbar ablaufen zu lassen.

Hierzu kann durch den Einsatz des Zufallszahlengenerators (= des sogenannten "R[andom] N[umber]G[enerator]") in erfindungswesentlicher Weise eine zur gewünschten Aktion führende Instruktionsfolge aus einer Vielzahl möglicher Instruktionsfolgen ausgewählt werden. Da mehrere verschiedene Instruktionsfolgen zum selben Ergebnis führen, kann der äußere Beobachter die laufende Aktion des Mikrocontrollers nicht als Folge der ausgewählten Instruktionsfolge nachvollziehen bzw. analysieren. Durch einen derartigen zufälligen Programmablauf werden erfindungsgemäß Rückschlüsse auf verarbeitete Daten erheblich erschwert bzw. ganz verhindert.

Durch die Hardwareimplementation des Mikrocontrollers sowie durch das zugeordnete Verfahren gemäß der vorliegenden Erfindung wird demzufolge weniger das Beobachten als vielmehr das Verstehen sowie Analysieren des internen Programmablaufs auf dem Mikrocontroller erschwert. In diesem Zusammenhang wird davon ausgegangen, dass es dem unautorisierten bzw. unbefugten Beobachter durchaus gelingen kann, Informationen zum ausgeführten Code zu erhalten.

Wesentlicher Bestandteil der vorliegenden Erfindung ist nun die Möglichkeit, Sprünge bzw. Verzweigungen im Programm unabhängig von internen Zuständen der Software in zufälliger Weise vornehmen zu können. Die Hardware des Mikrocontrollers bietet hierbei zusammen mit dem vorgesehenen Hardware-Zufallszahlengenerator die Möglichkeit, einen Programmsprung bzw. eine Programmverzweigung in Abhängigkeit vom Zustand des Zufallszahlengenerators durchzuführen oder abzulehnen. Die Zustände und die Werte dieses Zufallszahlengenerators sind nach außen hin nicht sichtbar.

Gemäß einer besonders erfinderischen Weiterbildung kann eine gleiche Funktionalität von Programmsprüngen bzw. von Programmzweigen durch Ausführen verschiedener, unterschiedlich implementierter Programmsprünge bzw. Programmzweige erreicht werden, das heißt bei gleicher Funktion liegt eine unterschiedliche Kodierung vor. Alternativ oder in Ergänzung hierzu kann auch gezielt eine unterschiedliche Funktionalität von Programmsprüngen bzw. von Programmzweigen herbeigeführt werden.

Gemäß einer bevorzugten Ausgestaltung der vorliegenden Erfindung ergibt sich eine weitere Verbesserung des Unsichtbarmachens von bedingten Sprüngen bzw. von konditionellen Verzweigungen dann, wenn Vor- und Rücksprünge bzw. Vor- und Rückverzeigungen kombiniert werden, so daß sich relativ schnell eine sehr hohe Anzahl von unterschiedlich implementierten Programmsprüngen bzw. Programmzweigen ergibt, die erfindungsgemäß in zufälliger Weise ausgewählt und ausgeführt werden können; so ergeben sich im exemplarischen Falle eines binären Baums mit Vorwärtssprüngen zum Beispiel sechzehn Sprünge, das heißt $16^4 = 65.536$ Möglichkeiten, das Programm unterschiedlich auszuführen.

Für einen äußeren Beobachter zeigt der Programmablauf erfindungsgemäß ein unvorhersehbares und nicht reproduzierbares Verhalten. Da aus einem derartigen Programmablauf bei großer Sprung- bzw. Verzweigungsanzahl nicht auf interne Zustände oder Daten des Mikrocontrollers geschlossen werden kann, wird mit dem Verfahren gemäß der vorliegenden Erfindung eine wirkungsvolle Methode bereitgestellt, diese Zustände und/oder Daten vor einem unautorisierten bzw. unbefugten Beobachter verborgen zu halten; dies resultiert in einer gesicherten Operation von Mikrocontrollern, insbesondere von SmartCard-Controllern, vor allem bei bedingten Programmsprüngen bzw. bei konditionellen Programmverzweigungen.

In zweckmäßiger Weise ist die hardwaremäßige Implementierung des Mikrocontrollers mit Zufallszahlengenerator auf vielfältige Art und Weise möglich, wobei zum Durchführen des Verfahrens gemäß der vorliegenden Erfindung vier wesentliche Implementierungsmethoden unabhängig voneinander oder in Kombination miteinander besonders empfehlenswert sind:

(i) Lesen der vom Zufallszahlengenerator erzeugten Zufallszahl per Software via Register und anschließendes Auswerten der gelesenen Zufallszahl mit bedingtem Programmsprung bzw. mit konditioneller Programmverzweigung;

(ii) bei Anordnung mindestens eines insbesondere bit-adressierbaren Zufallszahlenregisters (= sogenanntes R[andom]N[umber]R[egister]) im Mikrocontroller: Testen auf ein einzelnes Bit des Zufallszahlenregisters und konditionelles Verzweigen;

(iii) Implementieren eines entsprechenden Assemblerbefehls "branch on random bit" (= "Zweig auf Zufallsbit"), wobei ein festgelegtes Bit des Zufallszahlenregisters direkt an den Bedingungseingang für den bedingten Sprung bzw. für die konditionelle Verzweigung gelegt wird
(= komfortabelste und schnellste Implementierung mit geringstem Softwareaufwand); und/oder

(iv) als Variante zur unter Punkt (iii) beschriebenen Methode:
temporäres Ersetzen eines A[rithmetic]L[ogic]U[nit]-flag (A[rithmetic] L[ogic] U[nit] = logische Recheneinheit, wie sie in Mikrocontrollern vorhanden ist), das üblicherweise bedingte Sprünge bzw. konditionelle Verzweigungen steuert; durch ein Bit des Zufallszahlenregisters; das Ersetzen des ALU-flag kann softwaremäßig erfolgen, die dem ALU-bit entsprechenden bedingten Sprünge bzw. konditionellen Verzweigungen werden dann durch ein Bit des Zufallszahlenregisters gesteuert; das ALU-flag steht in dieser Zeit dann nicht für bedingte Sprünge bzw. für konditionelle Verzweigungen zur Verfügung.

Zusammenfassend sind bei der vorliegenden Erfindung also erhebliche Vorteile in den erheblich erschwerten Möglichkeiten der Analyse der internen Zustände oder Daten bei bedingten Sprüngen bzw. bei konditionellen Verzweigungen zu sehen. Mithin führt die vorliegende Erfindung unabhängig von der Struktur des (Mikrocontroller-)Programms auch stets zu denselben dynamischen Stromwerten und verhindert somit ein missbräuchliches sowie unerlaubtes Ausforschen zeitbedingter dynamischer Stromanalysen.

Die vorliegende Erfindung betrifft schließlich ein elektrisches oder elektronisches Gerät, gesteuert mittels mindestens eines Mikrocontrollers der vorstehend dargelegten Art.

Wie bereits vorstehend erörtert, gibt es verschiedene Möglichkeiten, die Lehre der vorliegenden Erfindung in vorteilhafter Weise auszugestalten und weiterzubilden. Hierzu wird einerseits auf die dem Anspruch 1 sowie dem Anspruch 5 nachgeordneten Ansprüche verwiesen, andererseits werden weitere Ausgestaltungen, Merkmale und Vorteile der vorliegenden Erfindung nachstehend anhand des durch Figur 1 veranschaulichten Ausführungsbeispiels näher erläutert.

Es zeigt:

Fig. 1 in schematischer Darstellung ein Blockschaltbild eines Ausführungsbeispiels eines mit dem Verfahren gemäß der vorliegenden Erfindung betriebenen Mikrocontrollers gemäß der vorliegenden Erfindung.

In Figur 1 ist ein Ausführungsbeispiel für einen als SmartCard-Controller ausgebildeten, zum Steuern eines elektrischen oder elektronischen Geräts vorgesehenen Mikrocontroller 100 dargestellt, dessen Programmierung in einer maschinenabhängigen Assemblersprache vorgenommen ist und abgearbeitet wird. Hierbei werden verfahrensgemäß die Assemblerbefehle mit Ausnahme bedingter Programmsprünge bzw. konditioneller Programmverzweigungen unabhängig von Daten ausgeführt.

Der Mikrocontroller 100 zeichnet sich nun dadurch aus, dass dem Mikrocontroller 100 ein Zufallszahlengenerator 10 zugeordnet ist, mittels dessen die Programmsprünge bzw. Programmverzweigungen abhängig vom Zustand des Zufallszahlengenerators 10 und unabhängig vom internen Zustand der Programmierung des Mikrocontrollers 100 ausgeführt werden können.

Mithin kann eine gleiche Funktionalität von Programmsprüngen bzw. von Programmzweigen durch Ausführen verschiedener, unterschiedlich implementierter Programmsprünge bzw. Programmzweige erreicht werden, das heißt bei gleicher Funktion liegt eine unterschiedliche Kodierung vor.

Hierzu wird die vom Zufallszahlengenerator 10 erzeugte Zufallszahl softwaremäßig via Register gelesen und anschließend mit bedingtem Programmsprung bzw. mit konditioneller Programmverzweigung ausgewertet. Alternativ oder in Ergänzung hierzu kann durch das Vorhandensein eines bit-adressierbaren, dem Zufallszahlengenerator 10 zugeordneten Zufallszahlenregisters 20 auf ein einzelnes Bit des Zufallszahlenregisters 20 getestet und bedingt gesprungen bzw. konditionell verzweigt werden.

Die komfortabelste und schnellste Implementierung mit dem geringsten softwaretechnischen Aufwand besteht darin, einen Assemblerbefehl ("branch on random bit" = "Zweig auf Zufallsbit") zu implementieren, wobei ein festgelegtes Bit des Zufallszahlenregisters 20 unmittelbar an den Bedingungseingang für den bedingten Sprung bzw. für die konditionelle Verzweigung gelegt wird.

Die Programmierung des Mikrocontrollers 100 erlaubt auch eine Variante hierzu, bei der ein bedingte Sprünge bzw. konditionelle Verzweigungen steuerndes A[rithmetic] L[ogic] U[nit]-flag softwaremäßig durch ein Bit des Zufallszahlenregisters 20 ersetzt wird, so dadd die dem A[rithmetic]L[ogic]U[nit]-Bit entsprechenden bedingten Sprünge durch das Bit des Zufallszahlenregisters 20 gesteuert werden.

Mittels des Mikrocontrollers 100 gemäß Figur 1 sowie mittels des Verfahrens zum Abarbeiten der Programmierung des Mikrocontrollers 100 kann eben diese auf dem Mikrocontroller 100 ablaufende Programmierung vollständig verschleiert werden, indem durch geeignetes Verarbeiten der vom Zufallszahlengenerator 10 erzeugten Zufallszahlen ein auf dem Mikrocontroller 100 ablaufendes Programm für einen äußeren Beobachter unvorhersehbar und nicht reproduzierbar abläuft.

Hierzu wird durch Einsatz des Zufallszahlengenerators 10 eine zur gewünschten Aktion führende Instruktion aus einer Vielzahl möglicher Instruktionen ausgewählt. Da mehrere verschiedene Instruktionen zum selben Ergebnis führen, kann der äußere Beobachter die laufende Aktion des Mikrocontrollers 100 nicht als Folge der ausgewählten Instruktion nachvollziehen bzw. analysieren. Durch einen derartigen zufälligen Programmablauf werden also Rückschlüsse auf verarbeitete Daten erheblich erschwert bzw. ganz verhindert.

BEZUGSZEICHENLISTE

100 Mikrocontroller, insbesondere SmartCard-Controller

10 Zufallszahlengenerator oder R[andom]N[umber]G[enerator]

20 insbesondere bit-adressierbares Zufallszahlenregister oder insbesondere bit-adressierbares R[andom]N[umber]R[egister]

PATENTANSPRÜCHE

1. Mikrocontroller (100), dessen Programmierung in mindestens einer maschinenabhängigen Assemblersprache vorgenommen ist, bei der die Assemblerbefehle mit Ausnahme bedingter Programmsprünge bzw. konditioneller Programmverzweigungen im wesentlichen unabhängig von Daten ausführbar sind,

gekennzeichnet durch

mindestens einen dem Mikrocontroller (100) zugeordneten Zufallszahlengenerator (10), mittels dessen die Programmsprünge bzw. Programmverzweigungen

- abhängig vom Zustand des Zufallszahlengenerators (10) und/oder
- unabhängig vom internen Zustand der Programmierung des Mikrocontrollers (100)

ausführbar sind.

2. Mikrocontroller gemäß Anspruch 1,

gekennzeichnet durch

mindestens ein dem Zufallszahlengenerator (10) zugeordnetes, insbesondere bitadressierbares Zufallszahlenregister (20).

3. Mikrocontroller gemäß Anspruch 1 oder 2,

gekennzeichnet durch

eine Ausgestaltung als SmartCard-Controller.

4. Elektrisches oder elektronisches Gerät, gesteuert mittels mindestens eines Mikrocontrollers (100) gemäß mindestens einem der Ansprüche 1 bis 3.

5. Verfahren zum Abarbeiten der in mindestens einer maschinenabhängigen Assemblersprache vorgenommenen Programmierung eines Mikrocontrollers (100), wobei die Assemblerbefehle mit Ausnahme bedingter Programmsprünge bzw. konditioneller Programmverzweigungen im wesentlichen unabhängig von Daten ausgeführt werden,

dadurch gekennzeichnet,

dass die Programmsprünge bzw. Programmverzweigungen

- abhängig vom Zustand mindestens eines Zufallszahlengenerators (10) und/oder
- unabhängig vom internen Zustand der Programmierung des Mikrocontrollers (100)

ausgeführt werden.

6. Verfahren gemäß Anspruch 5,

dadurch gekennzeichnet,

dass die vom Zufallszahlengenerator (10) erzeugte Zufallszahl softwaremäßig via Register gelesen wird und anschließend die gelesene Zufallszahl mit bedingtem Programmsprung bzw. mit konditioneller Programmverzweigung ausgewertet wird.

7. Verfahren gemäß Anspruch 5 oder 6,

dadurch gekennzeichnet,

dass bei Vorliegen mindestens eines insbesondere bit-adressierbaren Zufallszahlenregisters (20) auf ein einzelnes Bit des Zufallszahlenregisters (20) getestet und bedingt gesprungen bzw. konditionell verzweigt wird.

8. Verfahren gemäß mindestens einem der Ansprüche 5 bis 7,

gekennzeichnet durch

das Implementieren mindestens eines Assemblerbefehls ("branch on random bit" = "Zweig auf Zufallsbit"), wobei ein festgelegtes Bit des Zufallszahlenregisters (20) insbesondere unmittelbar an den Bedingungseingang für den bedingten Sprung bzw. für die konditionelle Verzweigung gelegt wird.

9. Verfahren gemäß mindestens einem der Ansprüche 5 bis 8,
dadurch gekennzeichnet,
dass mindestens ein bedingte Sprünge bzw. konditionelle Verzweigungen steuerndes A[rithmetic]L[ogic]U[nit]-flag insbesondere softwaremäßig durch mindestens ein Bit des Zufallszahlenregisters (20) ersetzt wird, so daß die dem A[rithmetic]L[ogic]U[nit]-Bit entsprechenden bedingten Sprünge durch das Bit des Zufallszahlenregisters (20) gesteuert werden.

10. Verwendung eines Mikrocontrollers (100) gemäß mindestens einem der Ansprüche 1 bis 3 und/oder eines Verfahrens gemäß mindestens einem der Ansprüche 5 bis 9 zum vollständigen Verschleiern der auf dem Mikrocontroller (100) ablaufenden Programmierung, so daß mindestens ein auf dem Mikrocontroller (100) ablaufendes Programm für einen äußeren Beobachter unvorhersehbar und nicht reproduzierbar ist.

ZUSAMMENFASSUNG

Mikrocontroller und zugeordnetes Verfahren zum Abarbeiten der Programmierung des Mikrocontrollers

Um einen Mikrocontroller (100), dessen Programmierung in mindestens einer maschinenabhängigen Assemblersprache vorgenommen ist, bei der die Assemblerbefehle mit Ausnahme bedingter Programmsprünge bzw. konditioneller Programmverzweigungen im wesentlichen unabhängig von Daten ausführbar sind, sowie ein Verfahren zum Abarbeiten der in mindestens einer maschinenabhängigen Assemblersprache vorgenommenen Programmierung des Mikrocontrollers (100) so weiterzuentwickeln, dass das auf dem Mikrocontroller (100) ablaufende Programm für einen äußeren Beobachter völlig geheim und unvorhersehbar, das heißt nicht reproduzierbar ist, wird vorgeschlagen, dass die Programmsprünge bzw. Programmverzweigungen

- abhängig vom Zustand mindestens eines Zufallszahlengenerators (10) und/oder
- unabhängig vom internen Zustand der Programmierung des Mikrocontrollers (100)

ausgeführt werden.

Fig. 1

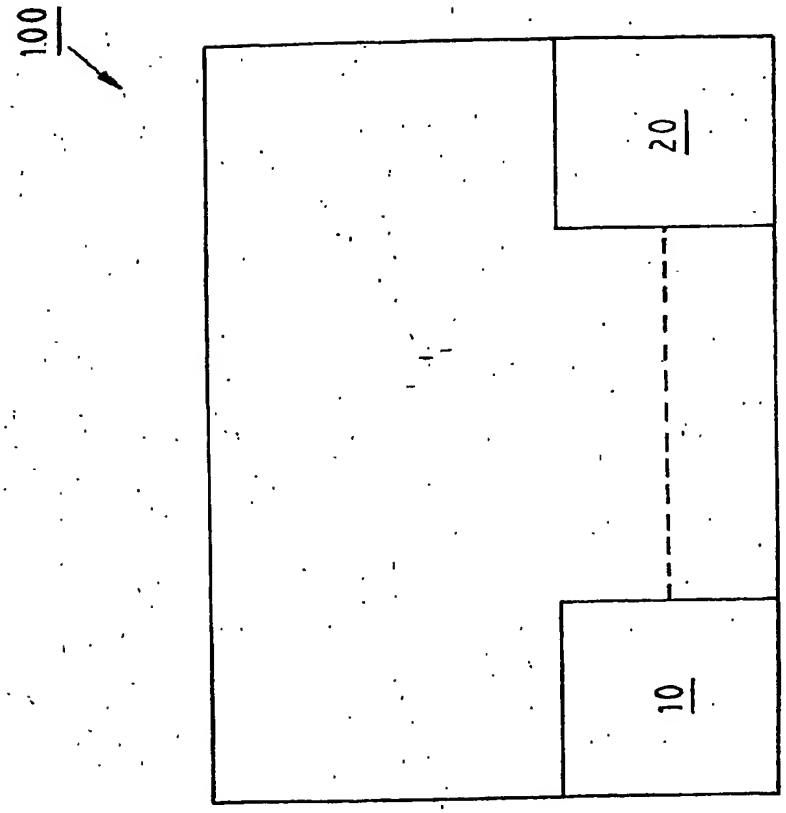

Fig.1

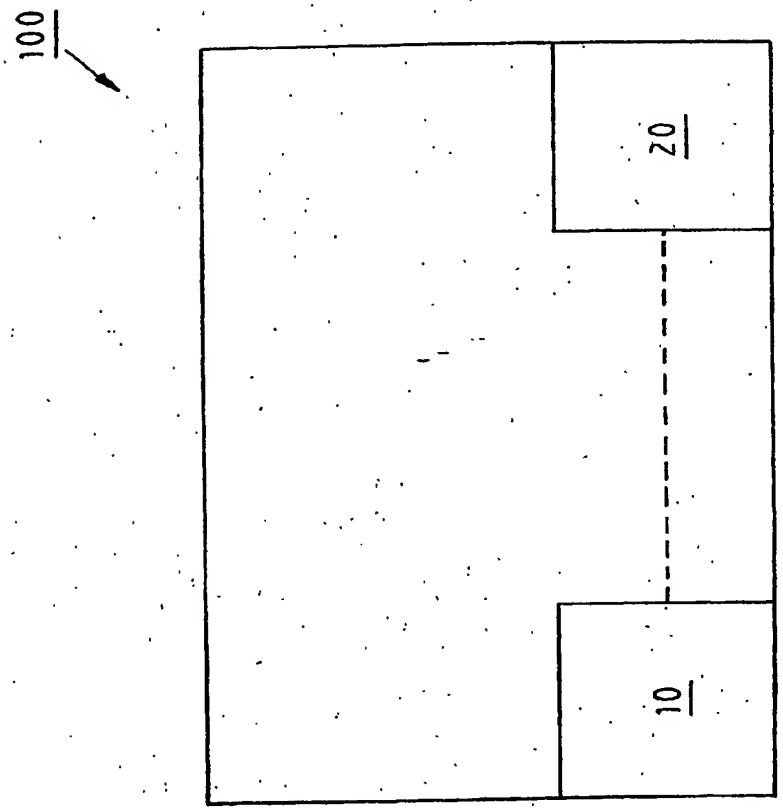

Fig.1